PWII.

Briefings.

Cada grupo deverá criar uma apresentação (slides, pôster, vídeo, etc.) de aproximadamente 15 a 20 minutos, abordando os seguintes tópicos, com foco na ferramenta designada: Azure DevOps.

Método e Metodologia.

O Azure DevOps é uma plataforma de gestão de ciclo de vida de aplicações (ALM) e de DevOps que fornece um conjunto integrado de serviços voltados ao desenvolvimento colaborativo e contínuo de software. Embora seja uma ferramenta flexível e configurável para diversos modelos de trabalho, sua estrutura base e seus modelos de processo são mais diretamente alinhados ao método **Scrum**, pois ele adota uma organização de artefatos e fluxos que refletem as práticas típicas dessa metodologia, como backlog de produto, sprints, planejamento de capacidade e acompanhamento por gráficos de burndown.

Contudo, o Azure DevOps oferece suporte nativo a diferentes **metodologias ágeis** por meio dos seus *process templates*, que definem a estrutura e os tipos de trabalho utilizados pelos times. Os principais são: o processo **Agile**, que combina práticas de Scrum, Kanban e XP e é o mais versátil; o processo **Scrum**, que segue mais fielmente os papéis e artefatos do framework Scrum; o processo **CMMI**, voltado a organizações que necessitam de rastreabilidade e governança mais formais, podendo ser usado em contextos híbridos ágeis; e o processo **Basic**, uma versão simplificada que privilegia o fluxo contínuo de trabalho, mais próximo do Kanban.

Além da aderência conceitual, o Azure DevOps disponibiliza recursos práticos que suportam a aplicação de princípios ágeis, como os *boards* configuráveis (para gestão visual de fluxo de trabalho e limitação de WIP), o gerenciamento de *user stories*, *tasks* e *bugs*, a execução de *sprints* e o planejamento iterativo. Também incorpora elementos de práticas do **Extreme Programming (XP)**, especialmente nas áreas de integração e entrega contínuas, testes automatizados e versionamento de código. Dessa forma, o Azure DevOps pode ser considerado uma plataforma que operacionaliza princípios ágeis em diferentes níveis de maturidade, com uma estrutura mais naturalmente orientada ao Scrum, mas com flexibilidade para suportar outras abordagens como Kanban e XP de forma nativa e integrada.

Kanbans.

O **Azure DevOps** implementa a visualização e a gestão de tarefas em formato **Kanban** por meio do módulo **Boards**, que oferece uma interface gráfica interativa para o acompanhamento do fluxo de trabalho. Cada **cartão** representa um item de trabalho (*Work Item*), como *User Stories*, *Tasks* ou *Bugs*, enquanto as **colunas** refletem os estágios do processo de desenvolvimento, podendo ser configuradas conforme o fluxo da equipe (por exemplo: *To Do*, *In Progress*, *Testing*, *Done*). A ferramenta permite definir **limites de WIP (Work in Progress)**, garantindo o controle de carga de trabalho e a identificação de gargalos.

Os **boards** são totalmente personalizáveis e permitem o uso de **Swimlanes** para segmentar tarefas por prioridade ou tipo, além de integrarem-se aos módulos de *Backlogs*, *Repos* e *Pipelines*, assegurando rastreabilidade do ciclo de vida do software. Um exemplo prático consiste na movimentação de um item de backlog, como “Implementar autenticação de usuários”, entre as colunas do board conforme sua evolução: *To Do* → *In Progress* → *Code Review* → *Testing* → *Done*. Esse processo visualiza e automatiza o fluxo de trabalho, fornecendo métricas como *Lead Time* e *Cycle Time*, fundamentais para análise de desempenho e melhoria contínua dentro das práticas ágeis.

(Projeto prático)

Workflows.

No **Azure DevOps**, os fluxos de processo (*workflows*) correspondem à definição formal dos estados pelos quais um item de trabalho (*Work Item*) transita durante seu ciclo de vida, permitindo que equipes de desenvolvimento acompanhem e controlem o progresso de forma estruturada e rastreável. Os workflows são configuráveis por meio do **Process Editor**, que possibilita definir não apenas os estados padrão, mas também estados intermediários, transições condicionais, campos obrigatórios e regras de automação específicas para cada tipo de item, como *User Story*, *Task* ou *Bug*.

Um fluxo típico utilizado no desenvolvimento de software segue a sequência **To Do** → **In Progress** → **Review** → **Done**, na qual *To Do* representa o trabalho planejado ainda não iniciado, *In Progress* indica a execução da tarefa, *Review* contempla etapas de validação, como *Code Review* ou testes automatizados, e *Done* marca a conclusão e entrega da funcionalidade. É possível estender esse fluxo com estados adicionais, como *Blocked*, quando uma tarefa depende de outra atividade, ou *Testing* e *Deployed*, permitindo maior granularidade na gestão do ciclo de desenvolvimento e maior aderência a práticas de DevOps.

Esses workflows se integram diretamente aos **Boards Kanban** e às **Sprints**, de modo que qualquer mudança de estado é refletida visualmente no quadro, permitindo monitoramento em tempo real, análise de métricas como *Lead Time* e *Cycle Time* e identificação de gargalos no fluxo de trabalho. A configuração flexível dos workflows garante compatibilidade com diferentes metodologias ágeis, como **Scrum** — com estados como *New → Active → Resolved → Closed* — e **Kanban**, favorecendo o gerenciamento visual contínuo das tarefas e a adaptação às necessidades específicas de cada equipe ou projeto.

Perfis.

No **Azure DevOps**, a gestão de usuários é organizada em torno de perfis ou funções que refletem os papéis típicos em metodologias ágeis e práticas de DevOps, sendo os principais: **Product Owner**, responsável por gerenciar o backlog, priorizar funcionalidades e definir critérios de aceitação; **Scrum Master**, que facilita a execução das cerimônias, remove impedimentos e monitora o progresso das iterações; **Desenvolvedor**, que cria, implementa e testa funcionalidades; e **Stakeholder**, que acompanha o andamento do projeto e visualiza relatórios e dashboards sem alterar itens de trabalho críticos.

A ferramenta gerencia permissões e visualizações por meio de **grupos de segurança e políticas de acesso** associados a cada projeto e a cada tipo de recurso (*Repos*, *Boards*, *Pipelines*, *Artifacts*). Cada perfil tem permissões específicas: por exemplo, o Product Owner pode criar e editar *Work Items*, priorizar backlog e gerenciar sprints; o Scrum Master pode alterar o fluxo de trabalho, atribuir tarefas e gerar relatórios de desempenho; o Desenvolvedor possui permissões para atualizar estados de tarefas, realizar commits em repositórios e acionar pipelines de CI/CD; enquanto o Stakeholder tem acesso principalmente a **dashboards**, relatórios e boards em modo de leitura.

Além disso, o Azure DevOps permite **controle granular de permissões**, incluindo permissões por área, iteração e recurso específico, garantindo que cada usuário visualize e interaja apenas com os itens e funcionalidades compatíveis com seu papel. A visualização nos **Boards**, dashboards e backlogs também é adaptada a esses perfis, promovendo segurança, governança e eficiência na colaboração entre equipes multidisciplinares.

Funcionalidade e Aplicabilidade.

Descrição/Foco na Ferramenta.

Ferramentas.

O **Azure DevOps** é uma plataforma unificada para gestão do ciclo de vida de software (*Application Lifecycle Management – ALM*), oferecendo serviços integrados que suportam planejamento ágil, desenvolvimento colaborativo, integração contínua, entrega automatizada e análise de métricas. Seus principais serviços são **Overview, Boards, Repos, Pipelines, Test Plans** e **Artifacts**, cada um com funcionalidades específicas que se articulam para fornecer rastreabilidade e governança do projeto.

O serviço **Overview** centraliza a visão do projeto, oferecendo *dashboards* personalizáveis, *widgets* interativos e *wiki*. Os *dashboards* permitem agregar informações de múltiplos serviços, como status de *Work Items*, progresso de sprints e métricas de código, enquanto os *widgets* podem exibir dados dinâmicos provenientes de *Repos*, *Pipelines* ou integrações externas, como GitHub e Slack. A *wiki* funciona como repositório de documentação colaborativa, permitindo armazenar guias, manuais e instruções operacionais para toda a equipe.

O serviço **Boards** gerencia fluxos de trabalho e tarefas, incluindo *Work Items*, quadros Kanban, *Backlogs*, *Sprints* e *Queries*. *Work Items* permitem detalhar tarefas, bugs ou histórias de usuário com estados, responsáveis, comentários e campos personalizados. Os quadros Kanban oferecem visualização em colunas configuráveis (*To Do, In Progress, Review, Done*) e suporte a *Swimlanes* para segmentação de prioridades ou tipos de tarefa. *Backlogs* e *Sprints* permitem planejamento iterativo de entregas, enquanto *Queries* possibilitam relatórios customizados, extraindo informações sobre progresso, distribuição de tarefas ou métricas de desempenho.

O serviço **Repos** fornece controle de versão de código baseado em Git, com funcionalidades de gerenciamento de arquivos, commits, branches, *pull requests* e histórico de alterações. Ele permite colaboração simultânea em diferentes linhas de desenvolvimento (*branches*) e revisão estruturada de código via *pull requests*, integrando-se com pipelines de build e testes automáticos para garantir qualidade e rastreabilidade.

O serviço **Pipelines** automatiza processos de *Continuous Integration/Continuous Delivery (CI/CD)*, permitindo configurar *pipelines* de build, teste e deploy em múltiplos ambientes. É possível criar *environments* específicos para produção, homologação ou testes, reutilizar variáveis e segredos via *Library* e integrar scripts de automação. Os pipelines podem ser acionados por eventos em *Repos* ou integrações externas, promovendo automação avançada e consistência no ciclo de entrega.

O **Test Plans** oferece recursos para planejamento e execução de testes manuais ou automatizados, permitindo rastrear defeitos e validar funcionalidades em múltiplos ambientes. Já o **Artifacts** gerencia pacotes e dependências, possibilitando

armazenamento e distribuição de bibliotecas e componentes reutilizáveis de forma segura e versionada.

Além disso, o Azure DevOps suporta integrações nativas com ferramentas externas como GitHub, Slack, Microsoft Teams, Jira, Docker e Kubernetes, ampliando automação, comunicação e rastreabilidade. Métricas e relatórios incluem *Lead Time*, *Cycle Time*, *Burndown Charts*, *Velocity* e tendências do backlog, fornecendo visibilidade completa do desempenho da equipe e eficiência do processo.

Em resumo, o Azure DevOps centraliza planejamento ágil, desenvolvimento, versionamento de código, automação de CI/CD, testes, distribuição de pacotes e análise de métricas em uma única plataforma, permitindo que equipes de desenvolvimento e operações colaborem de forma estruturada e eficiente, com suporte a metodologias como Scrum, Kanban e práticas DevOps.

Aplicabilidade.

O **Azure DevOps** é indicado principalmente para projetos de desenvolvimento de médio a grande porte, em que há necessidade de integração entre múltiplas equipes, rastreabilidade completa do ciclo de vida do software e automação de processos de *CI/CD*. A plataforma é especialmente adequada para organizações que trabalham com metodologias ágeis, como **Scrum** e **Kanban**, mas também oferece flexibilidade para fluxos híbridos ou mais formais, como CMMI.

Por meio de seus serviços integrados — *Boards*, *Repos*, *Pipelines*, *Test Plans* e *Artifacts* — o Azure DevOps suporta o planejamento e acompanhamento detalhado de tarefas, versionamento de código colaborativo, automação de builds e deploys, execução de testes e distribuição de pacotes. Isso o torna apropriado para **projetos complexos**, com múltiplos desenvolvedores, times distribuídos e ciclos iterativos de entrega.

Apesar de ser robusto, a ferramenta também pode ser utilizada por **times pequenos** que busquem organização ágil e automação básica, mas seu potencial se manifesta plenamente em cenários com alta complexidade, grande volume de tarefas, necessidade de métricas analíticas (como *lead time*, *cycle time* e *burndown*) e integração contínua entre desenvolvimento, testes e operação. Além disso, os dashboards e quadros Kanban permitem uma **gestão visual do projeto**, facilitando monitoramento do progresso, identificação de gargalos e alinhamento entre diferentes áreas da equipe.

Em síntese, o Azure DevOps é recomendado para projetos que exigem **governança estruturada, colaboração multidisciplinar, automação de processos e análise contínua de desempenho**, sendo ideal para empresas que desejam escalar práticas ágeis e DevOps em ambientes complexos e distribuídos.

Gerenciamento e Controle de Riscos.

O **Azure DevOps** contribui significativamente para a identificação, registro e mitigação de riscos em projetos de desenvolvimento por meio de seus recursos de rastreabilidade, monitoramento e gestão de tarefas. A ferramenta permite criar **Work Items específicos para riscos**, nos quais é possível registrar detalhes como tipo de risco, probabilidade, impacto, responsável e plano de mitigação. Esses itens podem ser categorizados com **labels ou tags**, como *Blocked*, *High Risk* ou *Dependency*, facilitando a rápida identificação de obstáculos que podem comprometer o progresso do projeto.

Além disso, os **quadros Kanban** e **backlogs** permitem visualizar de forma clara quais tarefas ou funcionalidades estão associadas a riscos ou impedimentos, destacando-as com cores, indicadores ou colunas específicas. É possível também configurar **alertas e regras automáticas** que notificam responsáveis sempre que um item crítico muda de estado, como quando uma tarefa bloqueada permanece em *In Progress* por muito tempo.

Os **dashboards e widgets customizáveis** fornecem visão consolidada do status de riscos em toda a equipe ou projeto. Métricas como número de itens bloqueados, tempo médio de resolução de impedimentos ou tendências de backlog ajudam na análise contínua e na priorização de ações preventivas. Essa visibilidade integrada permite às equipes **mitigar riscos de forma proativa**, ajustando prioridades, alocando recursos adequados ou acionando processos de revisão antes que o impacto no cronograma ou na qualidade se torne crítico.

Em resumo, o Azure DevOps apoia a **gestão de riscos estruturada**, combinando registro formal, categorização visual, monitoramento em tempo real e integração com processos ágeis, garantindo que riscos sejam identificados, acompanhados e mitigados de forma contínua ao longo do ciclo de desenvolvimento.

Gerenciamento e Controle de Prazos.

O **Azure DevOps** facilita o acompanhamento e cumprimento de prazos de projetos de desenvolvimento por meio de funcionalidades integradas de **planejamento, monitoramento e análise de progresso**, alinhadas a metodologias ágeis. Os **Sprints** permitem definir períodos de trabalho com datas de início e término, associando tarefas específicas do *Backlog* a cada iteração. Isso proporciona visibilidade clara sobre o que deve ser entregue em cada ciclo e ajuda a equipe a manter o ritmo de desenvolvimento.

Os **Boards Kanban** e **Work Items** exibem o status de cada tarefa, permitindo identificar rapidamente itens atrasados ou bloqueados. É possível configurar **alertas**

automáticos e notificações para atividades críticas que excedam prazos ou que permaneçam em estados como *In Progress* por muito tempo.

Os **relatórios de burndown e velocity** fornecem métricas quantitativas sobre o progresso do sprint ou do projeto, indicando se a equipe está no ritmo planejado para concluir todas as tarefas dentro do período definido. Além disso, os **dashboards customizáveis** podem consolidar informações sobre datas de entrega, milestones, dependências e atrasos, permitindo monitoramento contínuo do cronograma e suporte à tomada de decisão.

Dessa forma, o Azure DevOps garante **gestão visual e analítica de prazos**, promovendo planejamento estruturado, acompanhamento em tempo real e mecanismos de alerta que auxiliam equipes a cumprir datas de entrega, antecipar problemas e ajustar prioridades para minimizar riscos de atrasos.

Definição do Projeto.

O projeto desenvolvido tem como finalidade a construção de uma aplicação web dinâmica utilizando as tecnologias **HTML**, **PHP** e **MySQL**, com foco na integração entre front-end, back-end e banco de dados relacional. A iniciativa busca demonstrar, de forma prática, como ocorre o fluxo completo de processamento de dados em uma aplicação web: desde a captura das informações pelo usuário, passando pela validação na camada de lógica, até o armazenamento e recuperação no banco de dados.

A aplicação inicia-se por meio de uma página de **autenticação de usuário**, desenvolvida em HTML e estilizada através de recursos básicos de interface. As informações inseridas no formulário de login (usuário e senha) são enviadas ao servidor através do método POST e processadas por um script PHP. Esse script realiza a conexão com o banco de dados MySQL utilizando comandos nativos da linguagem, como **mysqli_connect() e mysqli_query()**, garantindo uma comunicação eficiente com o SGBD. O sistema verifica, então, se os dados fornecidos correspondem a um registro existente no banco e, em caso de autenticação bem-sucedida, gera o redirecionamento do usuário para a página principal do projeto.

Após o login, o usuário tem acesso à página principal, cujo objetivo é possibilitar a manipulação de dados armazenados no banco. Dentro dessa estrutura, são aplicados os conceitos de **CRUD (Create, Read, Update, Delete)**, permitindo que o usuário registre novas informações, visualize registros existentes, atualize dados previamente cadastrados e, quando necessário, realize exclusões. Esse processo demonstra o funcionamento completo do ciclo de persistência de dados em aplicações web, evidenciando a interação entre o front-end (formulários), back-end

(regras de negócio em PHP) e banco de dados (comandos SQL executados no MySQL).

O banco de dados utilizado é previamente estruturado através de um script SQL, que define tabelas, atributos, tipos de dados e possíveis chaves primárias e estrangeiras. Esse planejamento garante conformidade com os princípios do **modelo relacional**, assegurando integridade referencial, organização das informações e otimização das consultas. Durante o projeto, o ambiente de desenvolvimento é configurado através de ferramentas como XAMPP, WAMP ou MAMP, responsáveis por simular localmente um servidor web Apache e um servidor de banco de dados MySQL, permitindo a execução do código PHP e a interação com o banco.

Em termos de aprendizado, o desenvolvimento do projeto proporciona ao aluno uma visão prática do ciclo de requisição e resposta HTTP. Ao enviar um formulário HTML, o servidor recebe a requisição, processa-a utilizando PHP e devolve uma resposta ao cliente (navegador). Esse fluxo evidencia como aplicações web funcionam internamente e demonstra a importância da segurança no tratamento de dados sensíveis, como credenciais de login.

Assim, o projeto consolida o entendimento dos fundamentos da programação web dinâmica, evidenciando a interação entre camadas, o uso de formulários, a manipulação de banco de dados e o emprego de uma arquitetura simples, porém funcional, que representa o modelo básico utilizado em sistemas de autenticação e gerenciamento de informações no ambiente corporativo e acadêmico.

Backlog do Produto.

O backlog do produto contém as principais histórias de usuário e funcionalidades planejadas para o desenvolvimento da aplicação web, priorizadas de acordo com sua importância no projeto.

➔ ID História do Usuário Critérios de Aceitação
➔ Prioridade

1. Como usuário, quero fazer login no sistema, para acessar minha conta. O sistema deve validar usuário e senha no banco de dados e permitir acesso apenas se forem válidos. **Alta**
2. Como usuário, quero cadastrar novos dados, para armazenar informações no banco de dados. O sistema deve permitir inserir novos registros via formulário. **Alta**
3. Como usuário, quero visualizar os dados cadastrados, para consultar as informações existentes. O sistema deve exibir uma lista com todos os registros do banco. **Média**

4. Como usuário, quero editar dados existentes, para corrigir ou atualizar informações. O sistema deve permitir edição e atualização dos registros salvos. **Média**
5. 5 Como usuário, quero excluir dados, para remover informações desnecessárias. O sistema deve permitir a exclusão de registros de forma segura. **Baixa**
6. 6 Como administrador, quero visualizar logs de acesso, para monitorar o uso da aplicação. O sistema deve armazenar e listar os logins realizados. **Baixa**

Sprints e Planejamentos.

O desenvolvimento do projeto foi dividido em três sprints principais, seguindo o método Scrum, com planejamento de tarefas e metas específicas para cada etapa.

Sprint 1 – Autenticação e Estrutura Inicial (1 semana)
- Criação do banco de dados MySQL e definição das tabelas.
- Desenvolvimento da página de login em HTML e CSS.
- Implementação da validação de usuário e senha utilizando PHP.
- Testes de autenticação e redirecionamento para a página principal.

Sprint 2 – CRUD de Registros (1 semana)
- Criação dos formulários de cadastro, edição e exclusão de dados.
- Implementação das funções PHP para inserir, atualizar e excluir registros.
- Exibição dos dados armazenados no banco (função READ).
- Validação e filtragem de dados para segurança das operações.

Sprint 3 – Interface e Melhorias (1 semana)
- Melhoria da interface e da navegação entre as páginas.
- Inserção de mensagens de feedback ao usuário (sucesso/erro).
- Testes completos do fluxo CRUD e correção de erros.
- Finalização da documentação e preparação da apresentação.

Execução e Reviews.

Execução:

Durante as sprints, as tarefas foram organizadas e acompanhadas pelo Azure DevOps Boards, utilizando colunas de status (To Do, In Progress, Testing e Done).

O código foi versionado no Azure Repos, garantindo controle de alterações, commits por sprint e rastreabilidade. Cada integrante do grupo atuou conforme seu papel: o Product Owner priorizou o backlog, o Scrum Master acompanhou o progresso e os desenvolvedores realizaram a implementação e os testes.

Os testes foram realizados localmente em ambiente simulado (XAMPP/WAMP), validando a integração entre HTML, PHP e MySQL, além de garantir o funcionamento correto do sistema de login e das operações CRUD.

Reviews:

Ao término de cada sprint, foi feita uma Sprint Review para validação das entregas:

- Sprint 1: Autenticação funcional e banco de dados configurado.
- Sprint 2: CRUD implementado com sucesso e interface básica criada.
- Sprint 3: Layout aprimorado, testes finais concluídos e documentação finalizada.

O projeto atendeu aos objetivos definidos no backlog, demonstrando uma aplicação web dinâmica e funcional, com integração entre as camadas de front-end, back-end e banco de dados.